Final Project

Group11

4/6/2021

#Introduction

The focus of our project is on predicting if the persons Income will be over 50k$ a year or under 50k$. The data used in this project was taken from the 1994 Census Database and was provided by the UCI Machine Learning Repository.This is a medium sized dataset with 32561 observations and 15 attributes. Of these 15 attributes 9 are categorical and 6 are numerical.

The dataset has various information about the person like there Age, sex, gender, Occupation, Education-level, hours per week, race, native country etc, these are used to predict if the person makes above or under 50k a year

This Project is completed in R and uses the packages ggplot2, plyr, dplyr, class, tree, randomForest, and ROCR. We used Decision Trees, Logistic Regression, and Random Forests to perform predictive modeling on the data. The quality model was Random Forests, followed by Logistic Regression and then Decision Trees. The fashions showed that Educationwas indeed a very essential predictor in determining whether or no longer an man or woman made extra than $50,000, as well as Capital Gain, Relationship, Age, and Occupation. Race, Sex, and Working Class were consisitently marked as predictors that had the least amount of have an effect on on Income. This document will consist of the step-by-step strategies We took to discover these conclusions and explanations on the concepts.

Nature of data: • Data Set Characteristics: Multivariate • Attribute Characteristics: Categorical and Numerical • Number of Records: 32561 • Number of Attributes: 15

Attributes Info:

```
Attribute         :        Information
```

1. Income : The Income of the person 1 for >50k and 0 for <50k.
2. Age : continous.
3. Workingclass : Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
4. Final_Weight : final weight continous.
5. Education : Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
6. Education_num : continuous.
7. Marital_Status : Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
8. Occupation : Tech-support, Craft-repair, Other-service, Sales, Exec-manAgerial, Prof-specialty,Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving,Priv-house-serv, Protective-serv, Armed-Forces.
9. Relationship : Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
10. Race : White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
11. Sex : Female, Male.
12. Capital_gain : continuous.

13. Capital_loss : continuous.
14. Hours_per_week : continuous.
15. Native_country : United-States, Cambodia, England, Puerto-Rico ,Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad & Tobago, Peru, Hong,Holand-Netherlands Poland, Jamaica,Vietnam,

#Pre-Processing Dataset

```
#the folllowing packAges will be needed in order to perform our analysis:

#install.packAges("ggplot2")
#install.packAges("plyr")
#install.packAges("dplyr")
#install.packAges("class")
#install.packAges("tree")
#install.packAges("randomForest")
#install.packAges("ROCR")
#install.packages(MASS)

#Load libraries
library(ggplot2)
library(plyr)
library(dplyr)
library(class)
library(tree)
library(randomForest)
library(ROCR)
library(caret)
library(corrplot)
library(RColorBrewer)
library(MASS)
```

After downloading the dataset from the UCI Machine Learning Repository, read the data into R and check the structure of it.

```
#Reading the data files
Income_data <- read.csv("Income_data.csv")

#Add column labels
Income_data.names <- c("Age",
                 "Workingclass",
                 "Final_Weight",
                 "Education",
                 "Education_num",
                 "Marital_Status",
                 "Occupation",
                 "Relationship",
                 "Race",
                 "Sex",
                 "Capital_gain ",
                 "Capital_loss",
```

```r
                  "Hours_per_week",
                  "Native_country",
                  "Income")
colnames(Income_data) <-Income_data.names

#Show dataset
str(Income_data)
```

```
## 'data.frame':    32560 obs. of  15 variables:
##  $ Age           : int  50 38 53 28 37 49 52 31 42 37 ...
##  $ Workingclass  : chr  " Self-emp-not-inc" " Private" " Private" " Private" ...
##  $ Final_Weight  : int  83311 215646 234721 338409 284582 160187 209642 45781 159449 280464 ...
##  $ Education     : chr  " Bachelors" " HS-grad" " 11th" " Bachelors" ...
##  $ Education_num : int  13 9 7 13 14 5 9 14 13 10 ...
##  $ Marital_Status: chr  " Married-civ-spouse" " Divorced" " Married-civ-spouse" " Married-civ-spouse
##  $ Occupation    : chr  " Exec-managerial" " Handlers-cleaners" " Handlers-cleaners" " Prof-specialty
##  $ Relationship  : chr  " Husband" " Not-in-family" " Husband" " Wife" ...
##  $ Race          : chr  " White" " White" " Black" " Black" ...
##  $ Sex           : chr  " Male" " Male" " Male" " Female" ...
##  $ Capital_gain  : int  0 0 0 0 0 0 0 14084 5178 0 ...
##  $ Capital_loss  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Hours_per_week: int  13 40 40 40 40 16 45 50 40 80 ...
##  $ Native_country: chr  " United-States" " United-States" " United-States" " Cuba" ...
##  $ Income        : chr  " <=50K" " <=50K" " <=50K" " <=50K" ...
```

We have done this project in two parts: 1. Data Visulaization 2. Data Modelling

first we will explore the data and will do necessary visualization to understand the data better. We will try to find correlations between the variables.

```r
##Data exploration

#checking the dimension of the dataset
dim(Income_data)
```

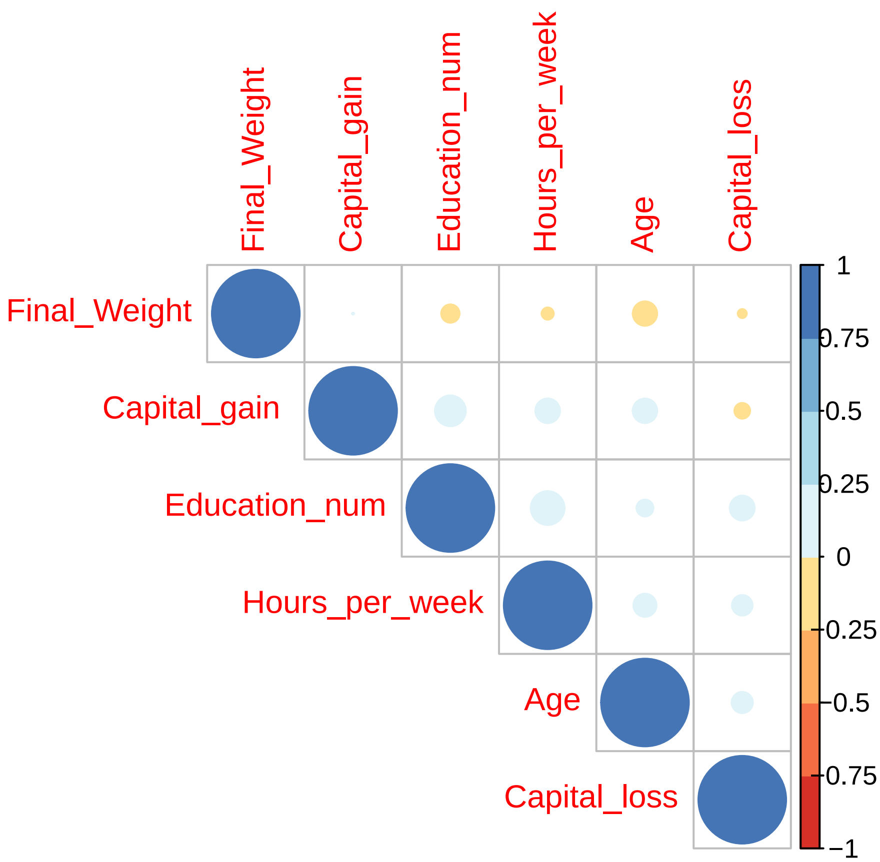
```
## [1] 32560    15
```

```r
#summary
summary(Income_data)
```

```
##       Age        Workingclass        Final_Weight      Education
##  Min.   :17.00   Length:32560       Min.   :  12285   Length:32560
##  1st Qu.:28.00   Class :character   1st Qu.: 117832   Class :character
##  Median :37.00   Mode  :character   Median : 178363   Mode  :character
##  Mean   :38.58                      Mean   : 189782
##  3rd Qu.:48.00                      3rd Qu.: 237055
##  Max.   :90.00                      Max.   :1484705
##  Education_num   Marital_Status      Occupation         Relationship
##  Min.   : 1.00   Length:32560       Length:32560       Length:32560
##  1st Qu.: 9.00   Class :character   Class :character   Class :character
##  Median :10.00   Mode  :character   Mode  :character   Mode  :character
##  Mean   :10.08
```

```
##   3rd Qu.:12.00
##   Max.   :16.00
##       Race                 Sex             Capital_gain     Capital_loss
##   Length:32560       Length:32560       Min.   :    0   Min.   :   0.00
##   Class :character   Class :character   1st Qu.:    0   1st Qu.:   0.00
##   Mode  :character   Mode  :character   Median :    0   Median :   0.00
##                                         Mean   : 1078   Mean   :  87.31
##                                         3rd Qu.:    0   3rd Qu.:   0.00
##                                         Max.   :99999   Max.   :4356.00
##   Hours_per_week  Native_country          Income
##   Min.   : 1.00   Length:32560       Length:32560
##   1st Qu.:40.00   Class :character   Class :character
##   Median :40.00   Mode  :character   Mode  :character
##   Mean   :40.44
##   3rd Qu.:45.00
##   Max.   :99.00
```

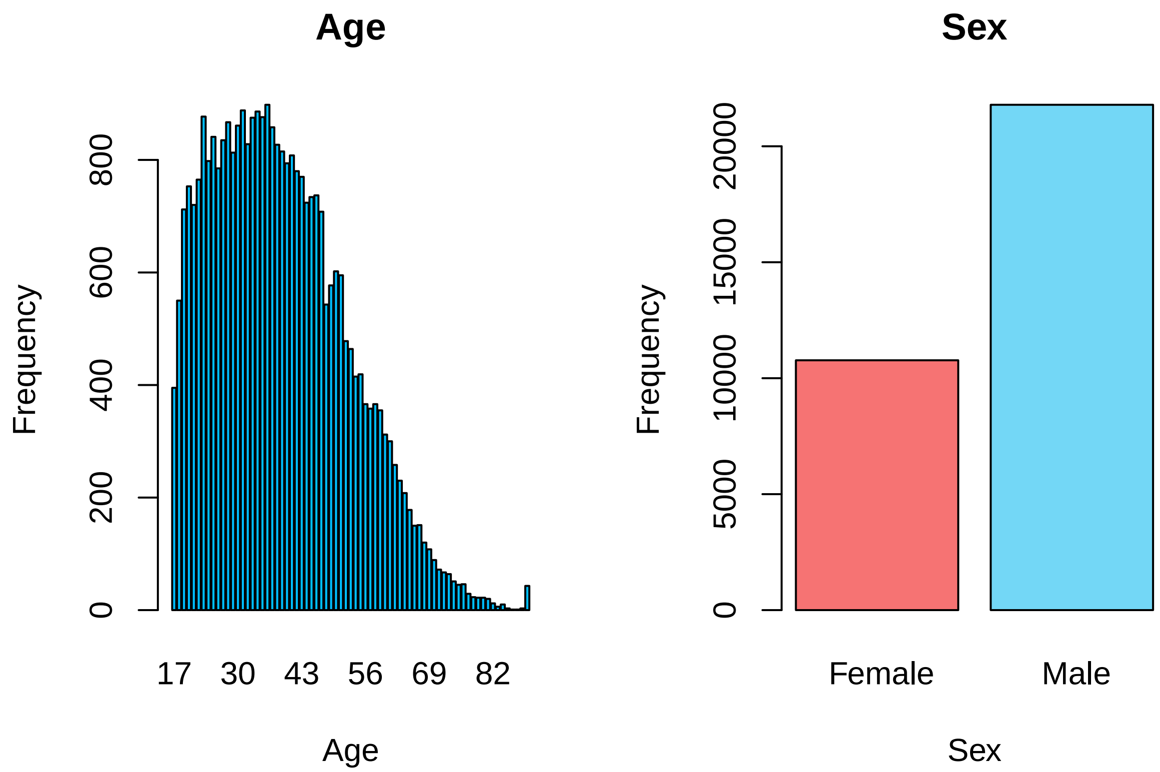
```
#Correlation plot
num.var <- c(1, 3, 5, 11:13)
corrplot(cor(Income_data[,num.var]),type="upper", order="hclust",
         col=brewer.pal(n=8, name="RdYlBu"))
```



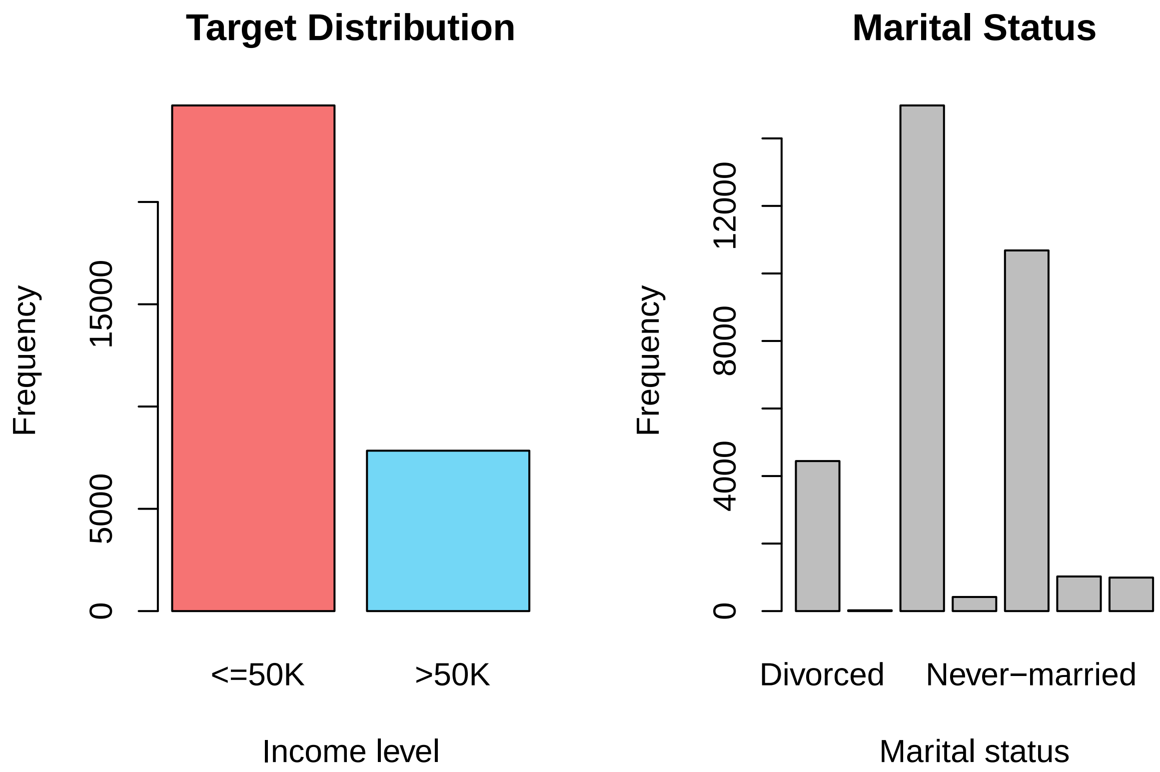
#Data Visualization

Visualizing the variables from the income data using bargraph and histogram.

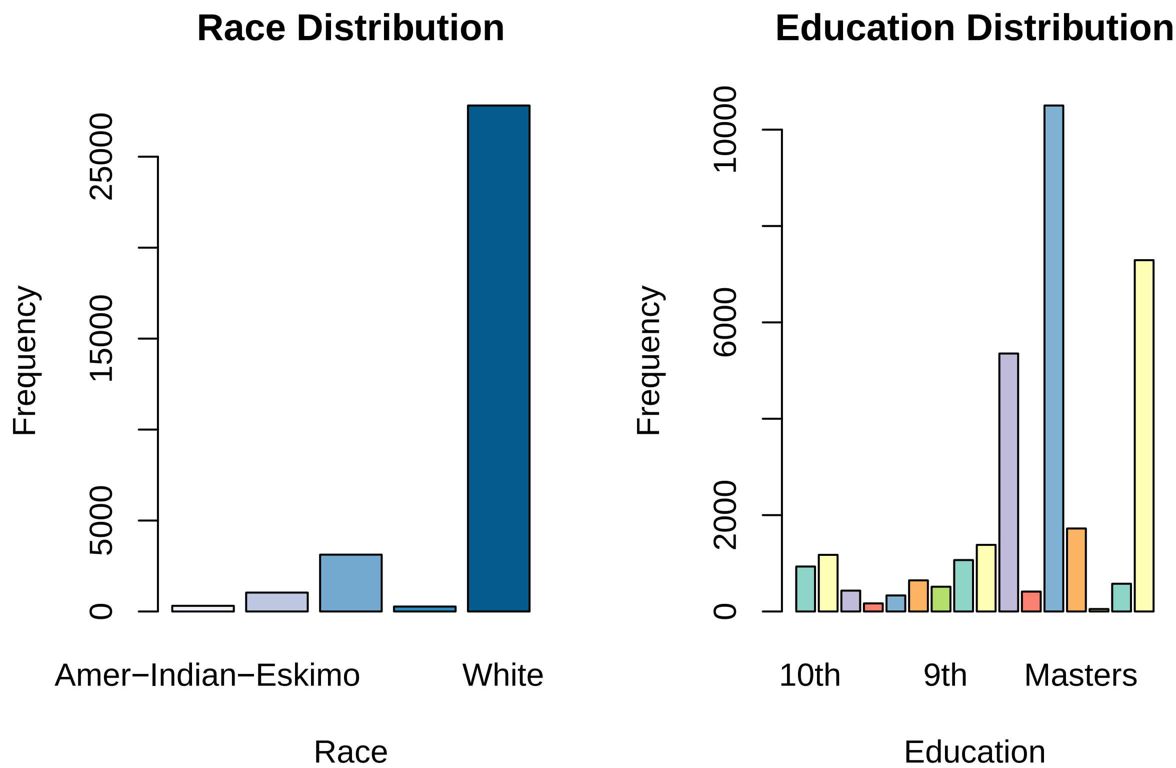
```
par(mfrow=c(1,2))
barplot(table(Income_data$Age), xlab = "Age", ylab = 'Frequency', main = "Age", col= "deepskyblue1")

barplot(table(as.factor(Income_data$Sex)), xlab = 'Sex', ylab = 'Frequency', main = 'Sex', col = c("#F67
```



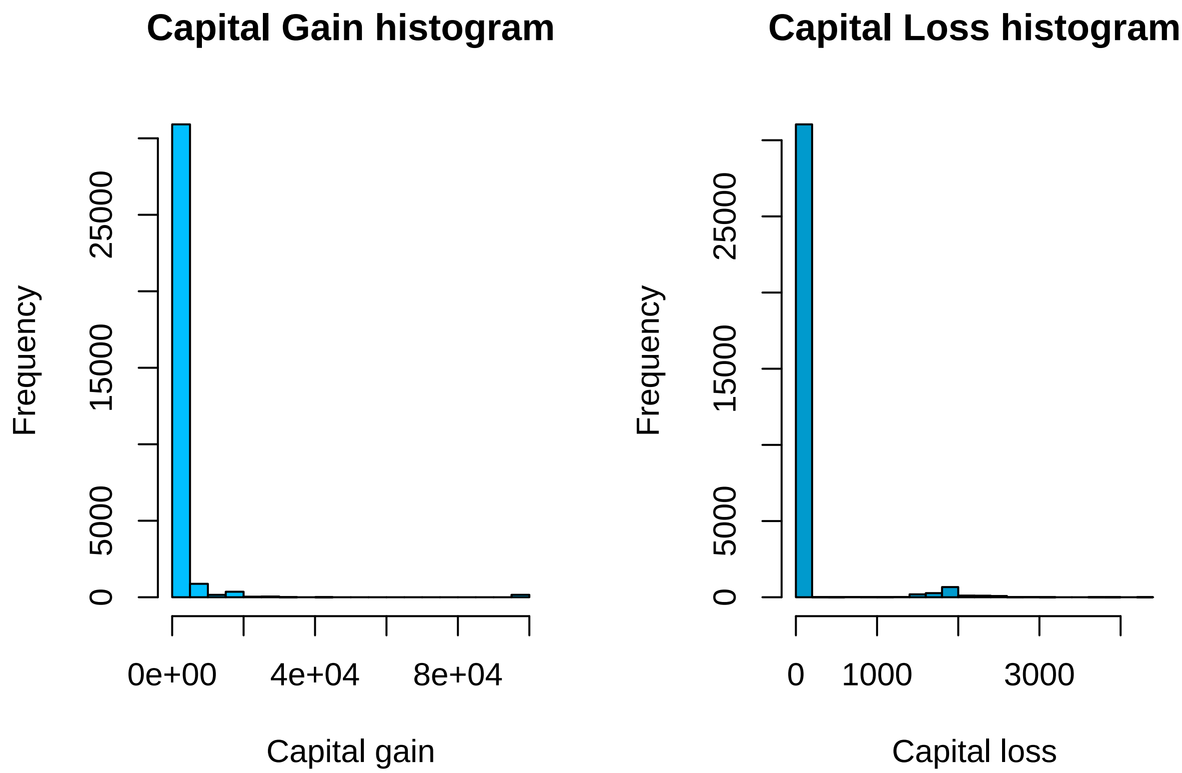

```
barplot(table(as.factor(Income_data$Income)), xlab = 'Income level', ylab = 'Frequency', main = 'Target

barplot(table(as.factor(Income_data$Marital_Status)), xlab = 'Marital status', ylab = 'Frequency',main =
```



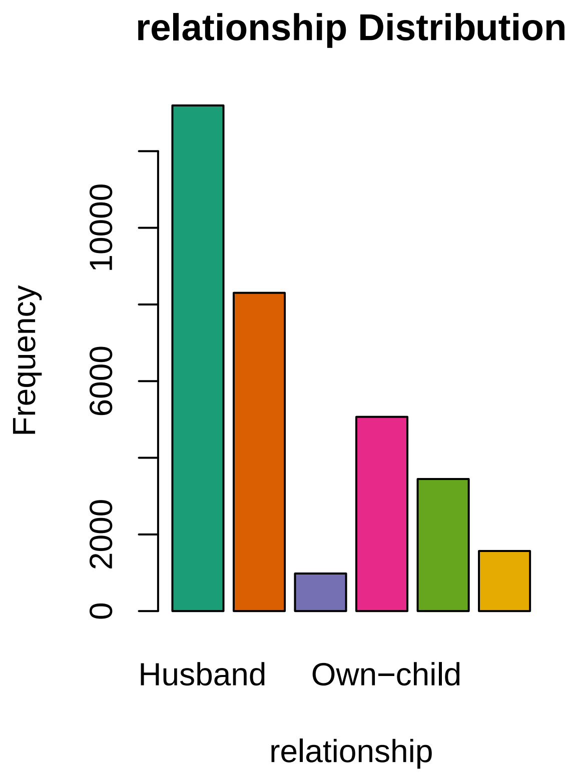

```
barplot(table(as.factor(Income_data$Race)), xlab = 'Race', ylab = 'Frequency', main = 'Race Distributio

barplot(table(as.factor(Income_data$Education)), xlab = 'Education', ylab = 'Frequency', main = 'Educati
```

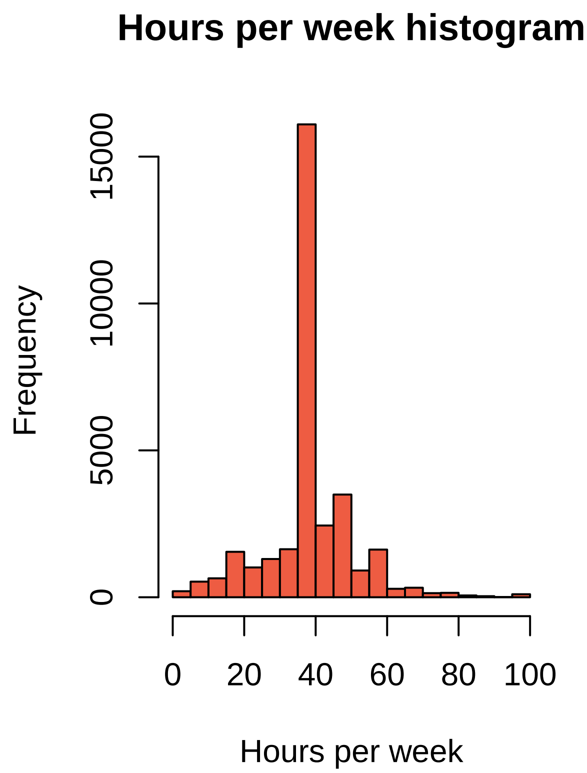



```
hist(Income_data$Capital_gain, xlab = 'Capital gain', ylab = 'Frequency', main = 'Capital Gain histogra

hist(Income_data$Capital_loss, xlab = 'Capital loss', ylab = 'Frequency', main = 'Capital Loss histogra
```




```
barplot(table(as.factor(Income_data$Relationship)), xlab = 'relationship', ylab = 'Frequency', main = 'r

hist(Income_data$Hours_per_week, xlab = 'Hours per week', ylab = 'Frequency', main = 'Hours per week his
```

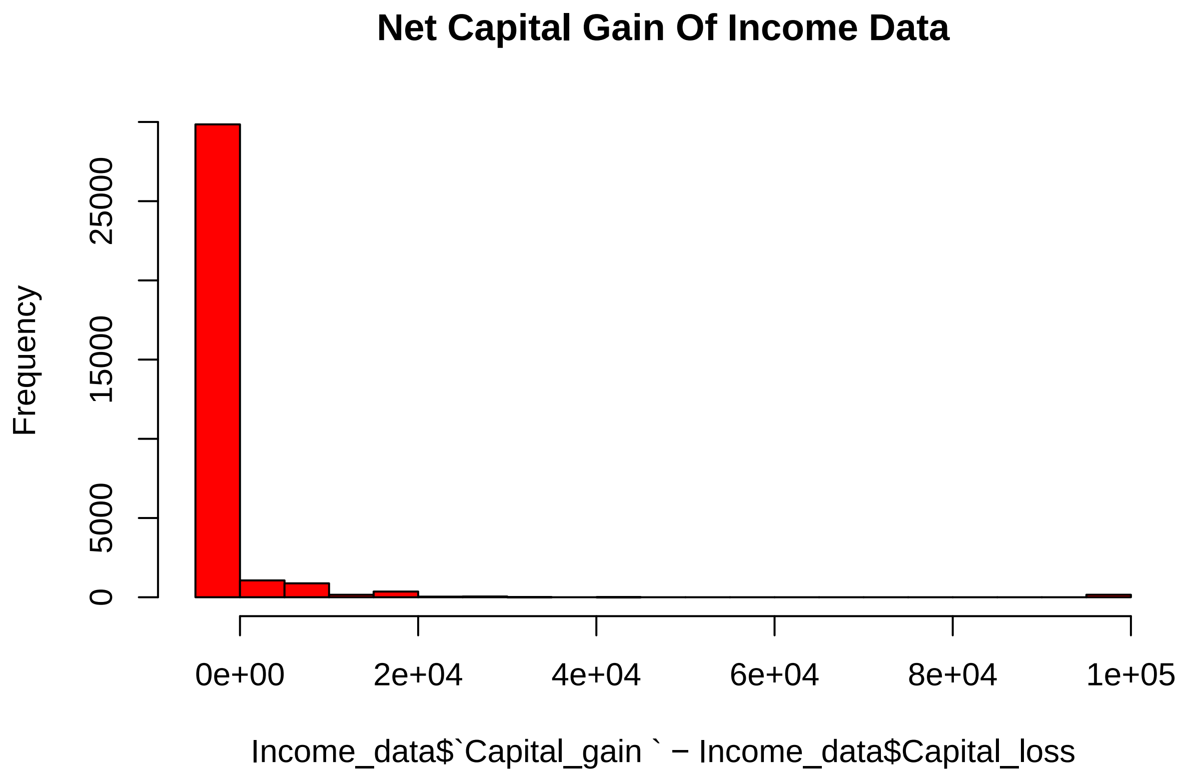





```
# checking for if someone has both capital gain and capital loss

sum(Income_data$Capital_loss > 0 & Income_data$`Capital_gain ` >0)
```

```
## [1] 0
```

```
#Net Capital Gain
hist(Income_data$`Capital_gain `- Income_data$Capital_loss , col = "Red" , main = "Net Capital Gain Of I
```



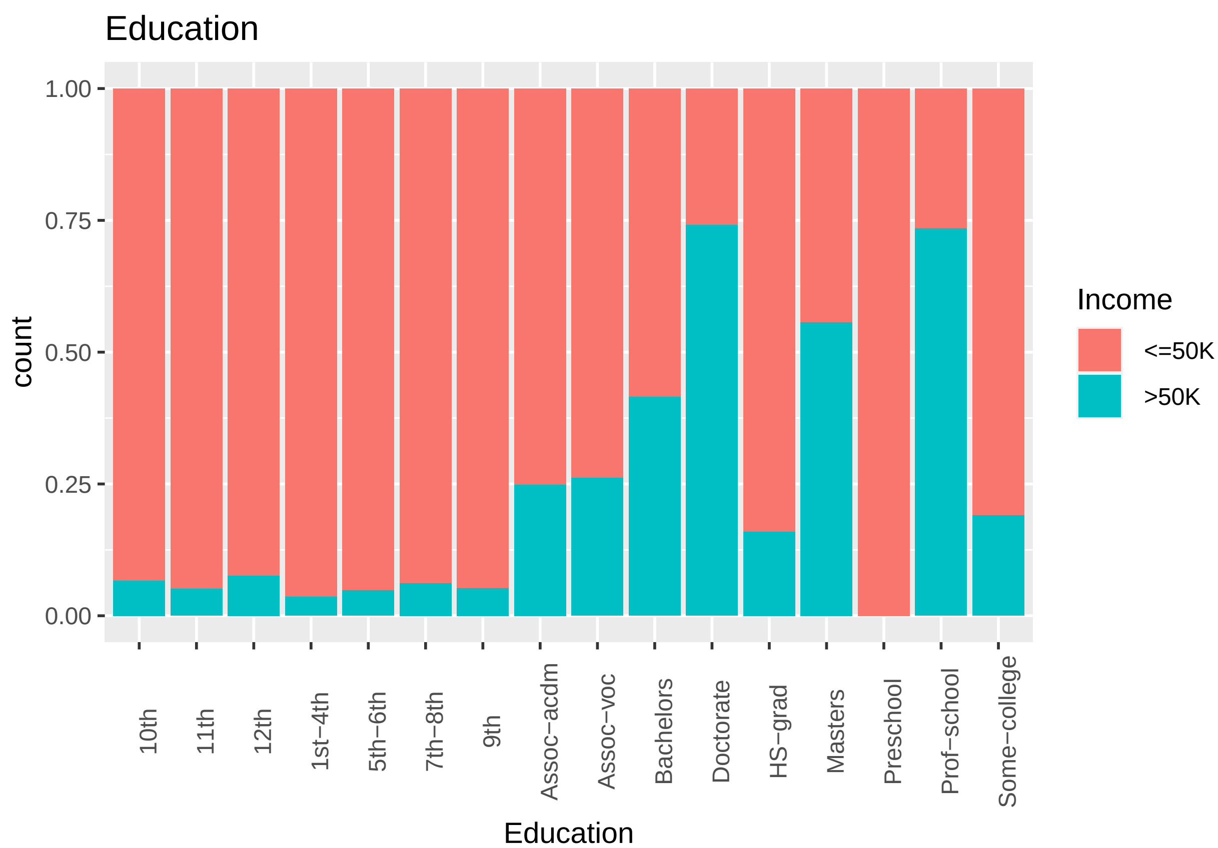

```
#workclass
ggplot(Income_data, aes(x = Workingclass, fill = Income)) + geom_bar(position="fill") + theme(axis.text
```


```
table(Income_data$Workingclass, Income_data$Income)
```

```
##
##                     <=50K   >50K
##    ?                 1645    191
##    Federal-gov        589    371
##    Local-gov         1476    617
##    Never-worked         7      0
##    Private          17733   4963
##    Self-emp-inc       494    622
##    Self-emp-not-inc  1817    724
##    State-gov          944    353
##    Without-pay         14      0
```

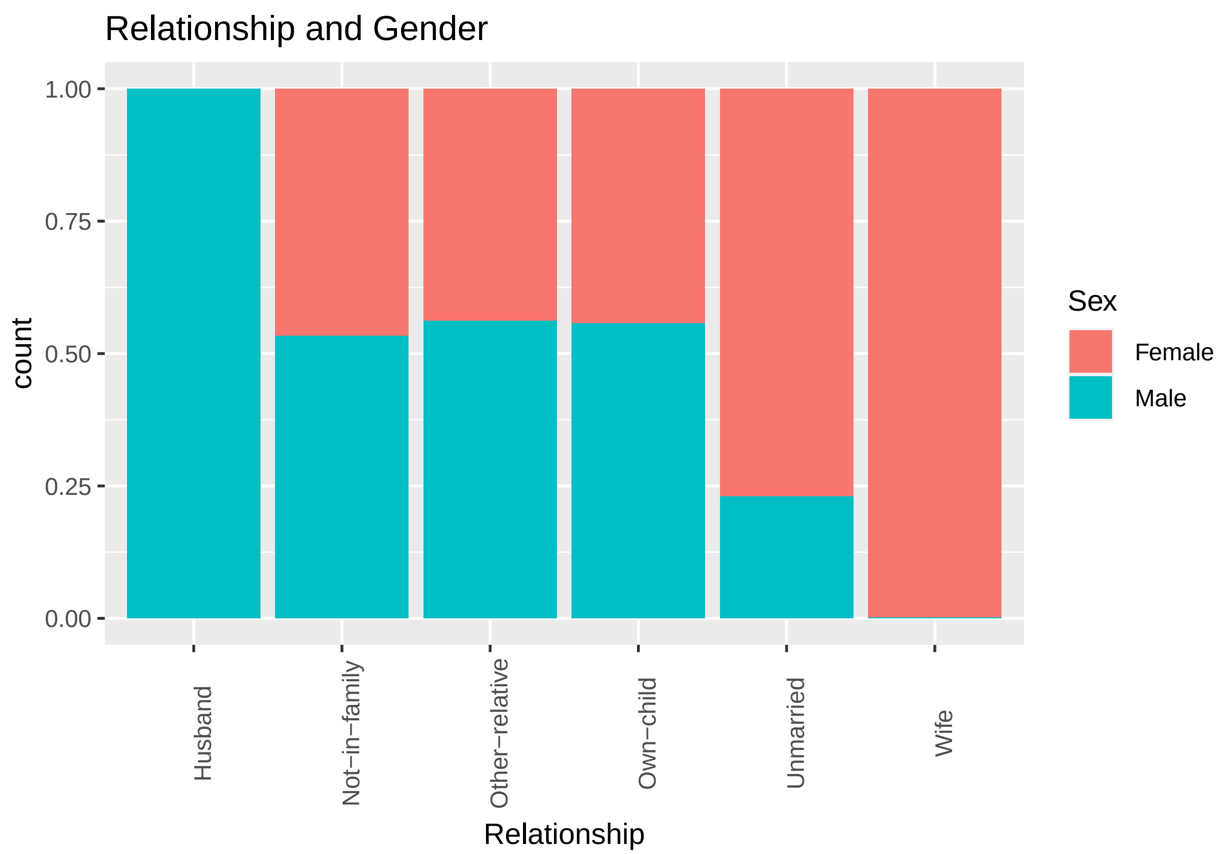
from the above graph we can see that: 0

```
#education

ggplot(Income_data, aes(x = Education, fill = Income)) + geom_bar(position="fill") + theme(axis.text.x
```



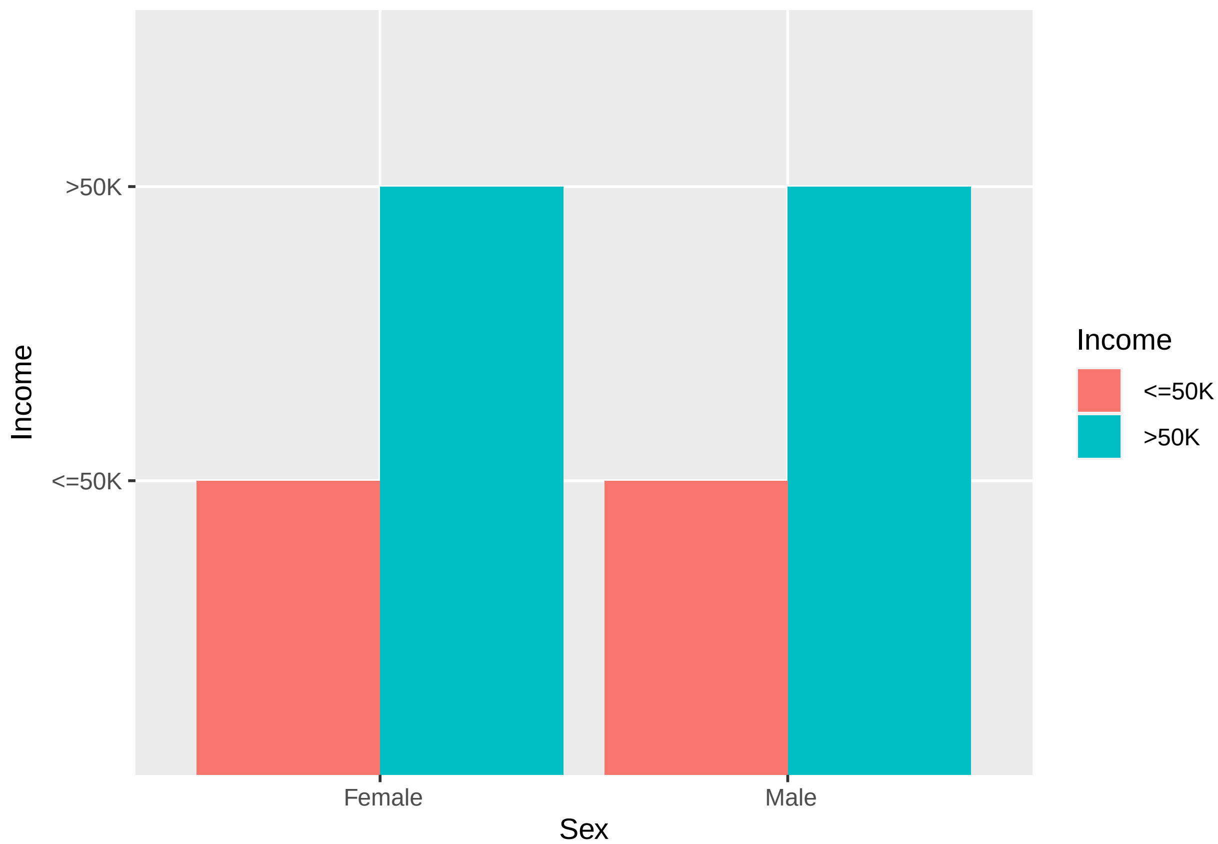
From the above graph we know what the education plays an important role in income.

```
#relationship vs sex
ggplot(Income_data, aes(x = Relationship, fill = Sex)) + geom_bar(position="fill") + theme(axis.text.x =
```

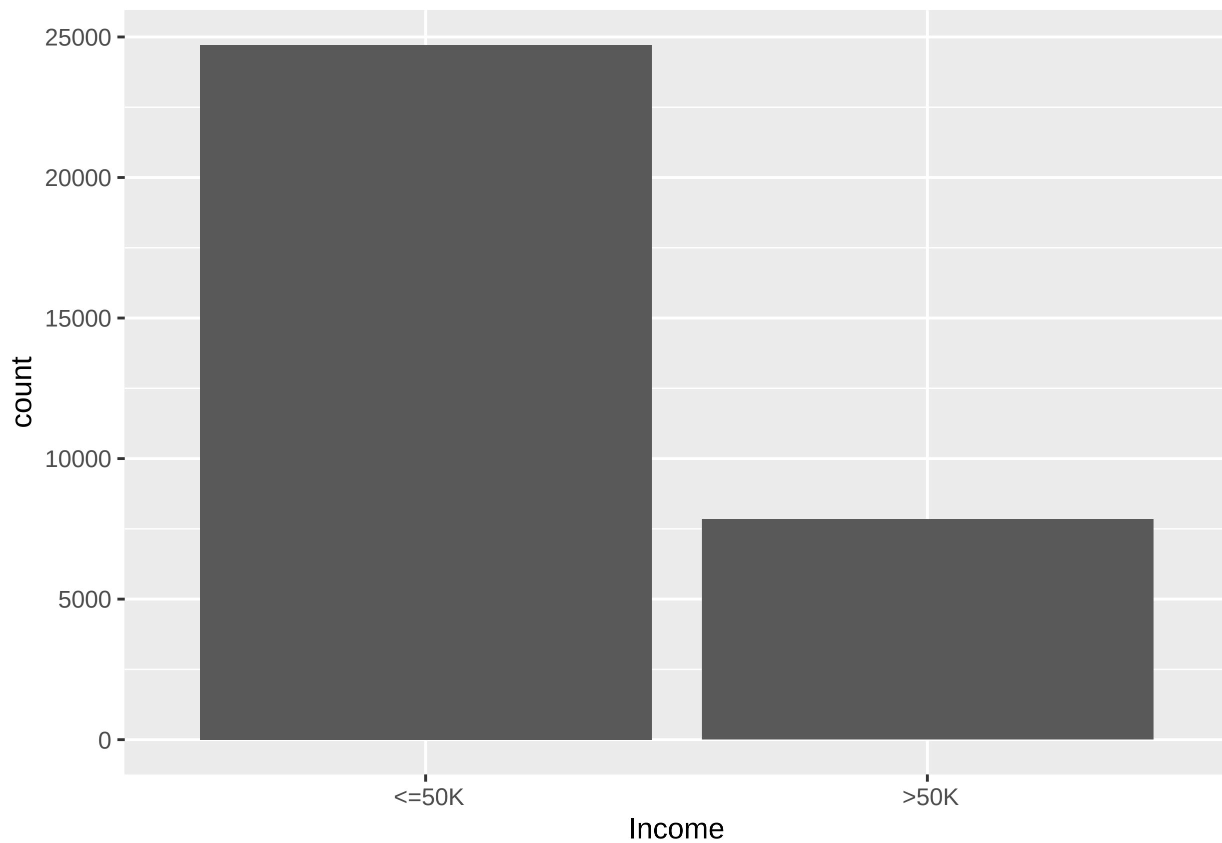


The original levels Husband Not-in-family Other-relative Own-child Unmarried Wife have been replaced by Not-in-family Other-relative Own-child Unmarried Spouse

```
ggplot(Income_data, aes(x=Sex  , y=Income)) +
  geom_bar(aes(fill = Income), stat="Identity", position="dodge")
```




```
ggplot(Income_data, aes(x=Income)) +
  geom_bar(stat="count")
```

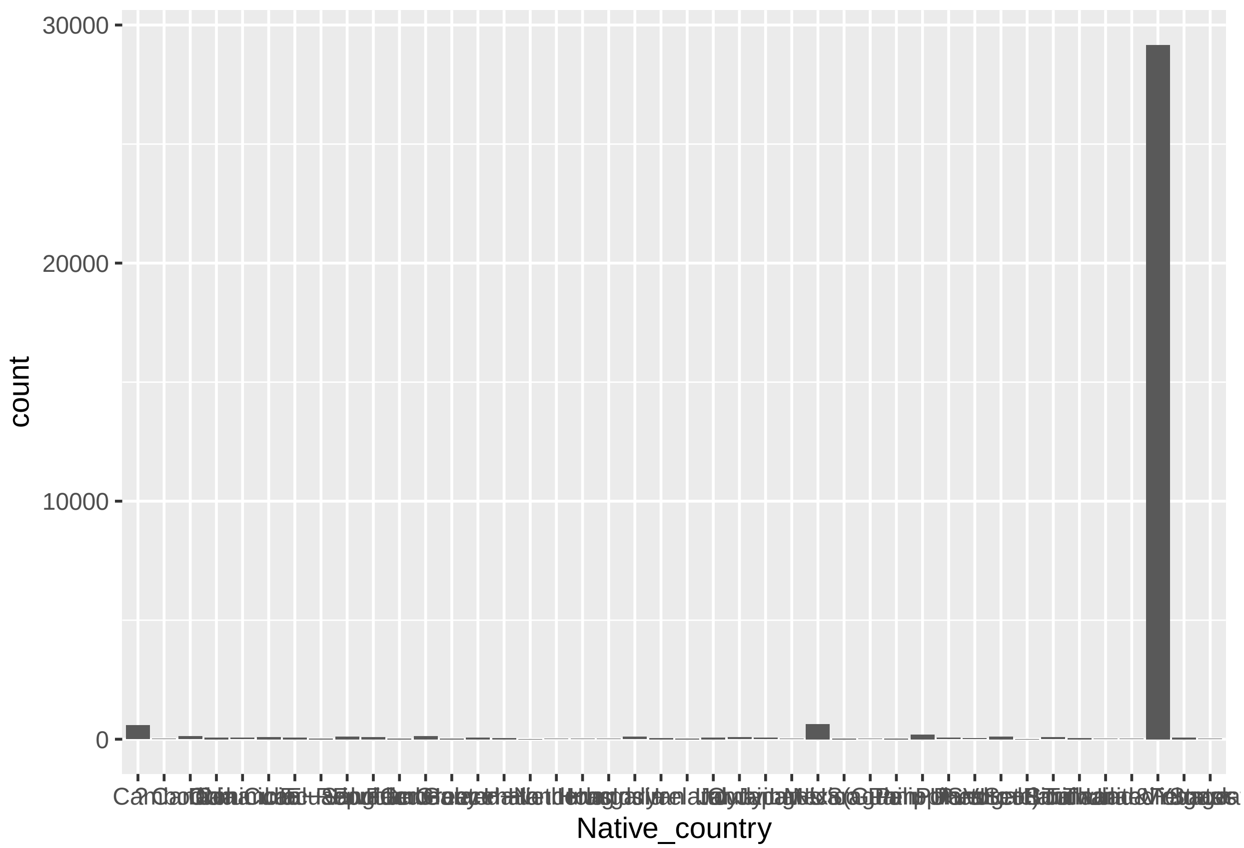



```
#Recode variables
Income_data$Income <- ifelse(Income_data$Income == " <=50K", 0, 1)
Income_data$Income <- as.factor(Income_data$Income)
```

Only about 1/4 of the observations have an Income value of ">$50K" that is 1. To solve this problem, we will under-sample the data, taking 4000 observations for our training set with equal amounts of randomly selected values for Income and 1000 randomly selected observations from the remainder of the data for the test set.

Removing variables

```
ggplot(Income_data , aes(x = Native_country)) +geom_bar()
```



We will also remove Native_country due to the fact that it will most likely not be a very meaningful predictor. Out of the 32560 observations, 90% have the value of "United States".

After inspecting the predictors Education_num and Education, we see that they are the portraying the same information. Education_num is just the numeric value of Education. We will keep Educationbecause of its interpretability and remove Education_num.

```
Income_data <- subset(Income_data , select = -c(Education_num , Native_country))
#View(Income_data)
```

#Removing Missing Values

We want to check how many missing values are in the dataset and then remove observations that have them.

```
#count missing values
Income_data[Income_data == " ?"] <- NA
sum(is.na(Income_data))
```

```
## [1] 3679
```

```
Income_data <- na.omit(Income_data)
Income_data <- data.frame(Income_data)

#'Re-factoring' variables to exclude the unwanted levels
Income_data$workclass <- as.factor(Income_data$Workingclass)
Income_data$occupation <- as.factor(Income_data$Occupation)
```

```r
#Check for class imbalance
summary(Income_data$Income)
```

```
##     0     1
## 23067  7650
```

```r
#Chi-Square Test
chisq.test(Income_data$Income, Income_data$Age)
```

```
##
##  Pearson's Chi-squared test
##
## data:  Income_data$Income and Income_data$Age
## X-squared = 3240.4, df = 71, p-value < 2.2e-16
```

```r
chisq.test(Income_data$Income, Income_data$Education)
```

```
##
##  Pearson's Chi-squared test
##
## data:  Income_data$Income and Income_data$Education
## X-squared = 4134, df = 15, p-value < 2.2e-16
```

```r
chisq.test(Income_data$Income, Income_data$Marital_Status)
```

```
##
##  Pearson's Chi-squared test
##
## data:  Income_data$Income and Income_data$Marital_Status
## X-squared = 6163.8, df = 6, p-value < 2.2e-16
```

```r
chisq.test(Income_data$Income, Income_data$Occupation)
```

```
##
##  Pearson's Chi-squared test
##
## data:  Income_data$Income and Income_data$Occupation
## X-squared = 3744.6, df = 13, p-value < 2.2e-16
```

```r
chisq.test(Income_data$Income, Income_data$Relationship)
```

```
##
##  Pearson's Chi-squared test
##
## data:  Income_data$Income and Income_data$Relationship
## X-squared = 6336.3, df = 5, p-value < 2.2e-16
```

```
chisq.test(Income_data$Income, Income_data$Race)
```

```
##
##  Pearson's Chi-squared test
##
## data:  Income_data$Income and Income_data$Race
## X-squared = 314.97, df = 4, p-value < 2.2e-16
```

```
chisq.test(Income_data$Income, Income_data$Sex)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  Income_data$Income and Income_data$Sex
## X-squared = 1440.6, df = 1, p-value < 2.2e-16
```

```
chisq.test(Income_data$Income, Income_data$Capital_gain)
```

```
##
##  Pearson's Chi-squared test
##
## data:  Income_data$Income and Income_data$Capital_gain
## X-squared = 5334.2, df = 117, p-value < 2.2e-16
```

```
chisq.test(Income_data$Income, Income_data$Capital_loss)
```

```
##
##  Pearson's Chi-squared test
##
## data:  Income_data$Income and Income_data$Capital_loss
## X-squared = 2354.3, df = 89, p-value < 2.2e-16
```

```
#chisq.test(Income_data$Income, Income_data$Hours.per.week)
```

By using chi-squared test we came to know that dependent variables of Income are all the other variables in the dataset except Final_Weight.The Final_Weight which is the final weight determined by the Census Organization is of no use in any of the analysis that we are doing henceforth and is removed. The educationnum if a repetitive variable which recodes the categorical variable Educationas a numeric variable but will be used in the analysis for decision trees, hence is not being removed.

```
#creating Training and Test Data
#set seed to ensure you always have same random numbers generated
set.seed(1)

#Separate values of Income
Income_data.GT50k <- subset(Income_data, Income_data$Income == 1)
Income_data.LT50k <- subset(Income_data, Income_data$Income == 0)

#Take 2000 random observations from both subsets of Income
Income_data.GT50k.indices <-sample(1:nrow(Income_data.GT50k),2000,replace = TRUE)
```

```
Income_data.LT50k.indices<-sample(1:nrow(Income_data.LT50k), 2000 , replace = TRUE)

#Combine subsets and randomize
Income_data.GT50k.train <- Income_data.GT50k[Income_data.GT50k.indices,]
Income_data.LT50k.train <- Income_data.LT50k[Income_data.LT50k.indices,]
Income_data.train <- rbind(Income_data.GT50k.train, Income_data.LT50k.train)
Income_data.train <- Income_data.train[sample(nrow(Income_data.train)),]

#Take row names from training observations
GT50k.rows <- row.names(Income_data.GT50k.train)
LT50k.rows <- row.names(Income_data.LT50k.train)
GT50k.rows <- as.numeric(GT50k.rows)
LT50k.rows <- as.numeric(LT50k.rows)

#Create subset of Income_data dataset without training observations
Income_data.sub <- Income_data[-GT50k.rows,]
Income_data.sub <- Income_data.sub[-LT50k.rows,]

#Take 1000 random observations for test set
set.seed(1)
test.indices <- sample(1:nrow(Income_data.sub), 1000 , replace = TRUE)
Income_data.test <- Income_data.sub[test.indices,]

summary(Income_data.train$Occupation)
```

```
##    Length     Class      Mode
##      4000 character character
```

```
summary(Income_data.test$Occupation)
```

```
##    Length     Class      Mode
##      1000 character character
```

For convenience purposes, we will create Xtrain, Ytrain, Xtest, and Ytest that containing the response and predictor variables for the training and test sets.

```
Ytrain <- Income_data.train$Income
Xtrain <- Income_data.train %>% filter(-Income)
Ytest <- Income_data.test$Income
Xtest <- Income_data.test %>% filter(-Income)
```

Now that we have concluded the pre-processing step, we can move on to creating models we will use to predict Income.

We will check the model accuracy with the confusion matrix: Accuracy: Accuracy is one of the most common metrics in measuring the classification models performance. It is defined as the ratio of number of correct predictions over the total predictions made. So, obviously the more theaccuracy is the better the model. We should always evaluate our models performance on the test data buton the train data since the model is built on the train data it usually performs better on the data it hasseen, but test data is something which the model has not seen so we can trust the test data's metrics. Accuracy = Number of correct predictions / Total number of predictions made

Confusion Matrix: We will use confusion matrix as our second metric as just in case if the accuracies of two models are relativelysame we look at confusion matrix to identify the false negatives and false positives. Here in our case theprimary goal is to predict if the person makes above 50k false positives and false negatives helps us in decidingthe best model.

Confusion matrix as the name says can be really confusing to understand, it is the summary of the predictionswhere the correct and incorrect classifications are summarized. It is has 4 elements • True positive - Observation is positive, and is predicted to be positive. • True Negative - Observation is negative, and is predicted to be negative.

##Methods 1. Decision Trees 2. Random Forest 3. Logestic Regression

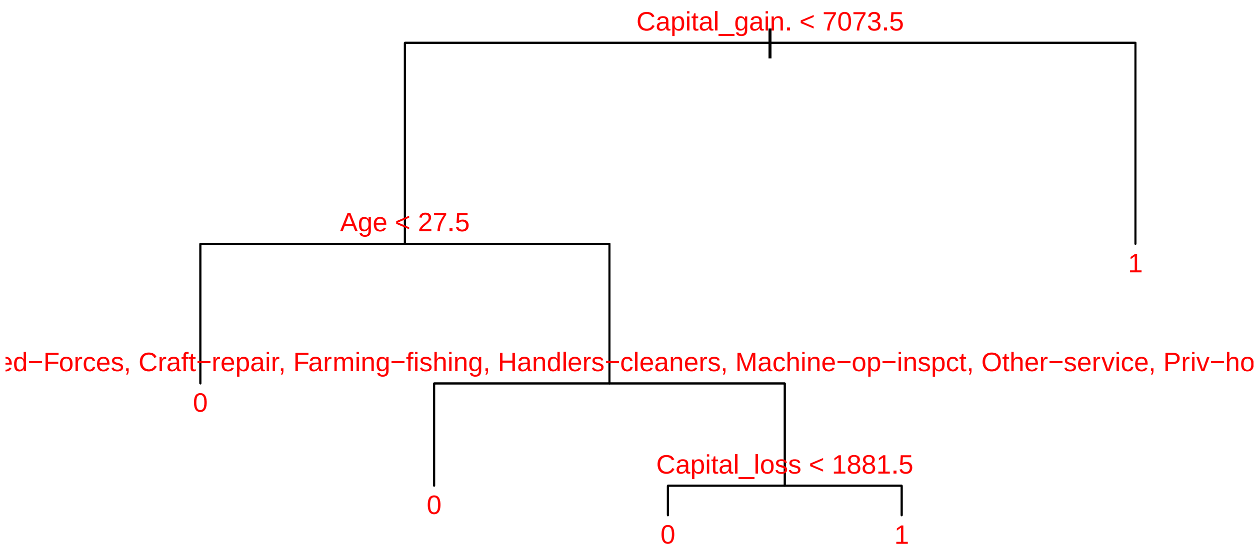
#Modelling

##Decision Tree A decision tree classifies by choosing a threshold on a feature and splits the data according to a 'splitting rule'. Since the features need to be numerical, we had to discard certain features and change how we represented others. For example, we could not convert Native_country into numerical values since this would cause an implicit feature ranking skewing our results.However, Educationis a feature that can be converted into a numerical value, as a certain level of Educationcan be higher or lower than others in rank. For this reason, we chose only to consider limited attributes (This is represented as a binary feature with 1 being male and 0 being female). The tree is then built on the training set and used to predict the binary value of the label (whether or not an individual makes more that $50,000) on the test set.

By using the tree() function, we are able to grow a tree on the training set, using Income as the response and all other variables as predictors

```r
set.seed(123)
erate <- function(predicted.value, true.value){ return(mean(true.value!=predicted.value))
}

#Fit tree on entire dataset
tree.full <- tree(Income ~ ., data= Income_data)
#Plot tree
plot(tree.full)
text(tree.full, pretty = 0, cex = .8, col = "red")
title("Classification Tree Built on Full Income_data Dataset")
```

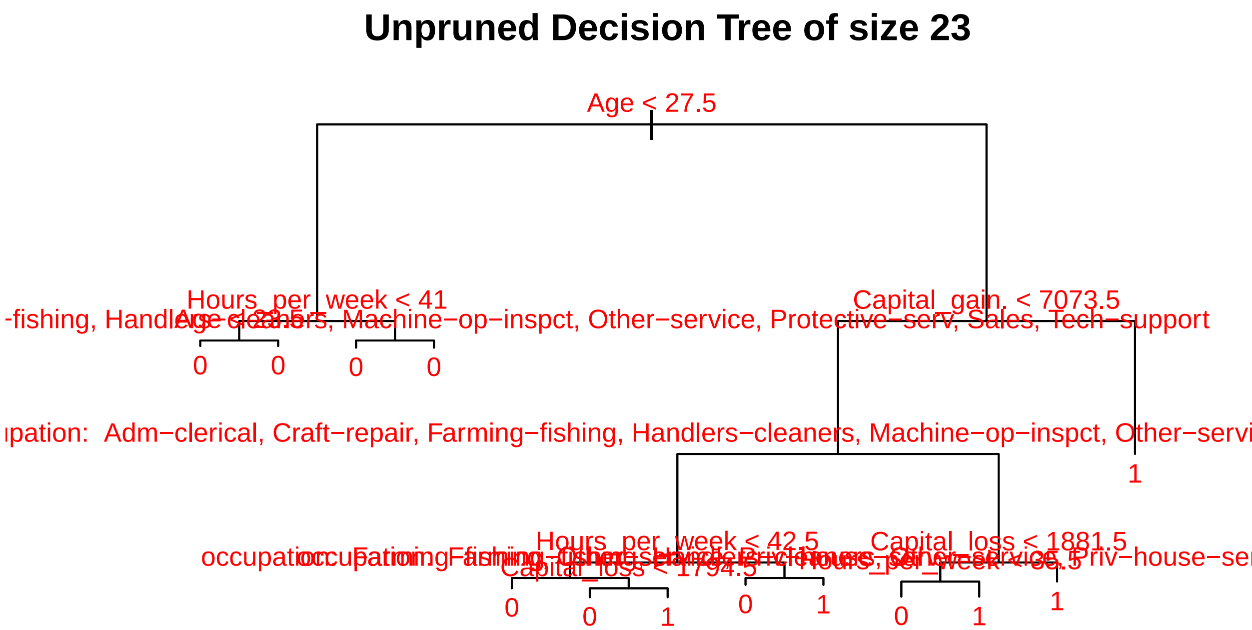
Classification Tree Built on Full Income_data Dataset




```r
#Fitting the model on training set
tree_Income_data <- tree(Income ~ . , data = Income_data.train)
tree_Income_data <- tree(Income ~ . ,
                         data = Income_data.train,
                         control = tree.control(4000,
                                                mincut = 5,
                                                mindev = 0.003))
```

Now that the tree has been created, we can now plot the tree to see what it looks like.

```r
#plotting the tree
plot(tree_Income_data)
text(tree_Income_data, pretty = 0, cex = .8, col = "red")
title("Unpruned Decision Tree of size 23")
```



Notice how at the split of each node, there is text describing the predictor variable and certain values within the variable. If an observation has these values, then it moves down the left side of the node. If it does not contain these values, it moves down the right side. The title of this tree is "Unpruned Decision Tree of size 23" because it is has 23 terminal nodes (the 1's and 0's at the bottom of the tree) and it is not pruned.

The next step is to prune our tree in order to find a better size and a better error rate. In order to prune the tree, we will perform a 10-fold cross-validation. This will allow us to find the best tree size that will minimize the error rate.

```
#Predict on training and test set
tree.pred.train <- predict(tree_Income_data, Income_data.train, type="class")
tree.pred.test <- predict(tree_Income_data, Income_data.test, type="class")

#Calculate train and test error on tree
tree.errors <- data.frame(train.error = erate(tree.pred.train, Ytrain),
                          test.error = erate(tree.pred.test, Ytest))
tree.errors
```

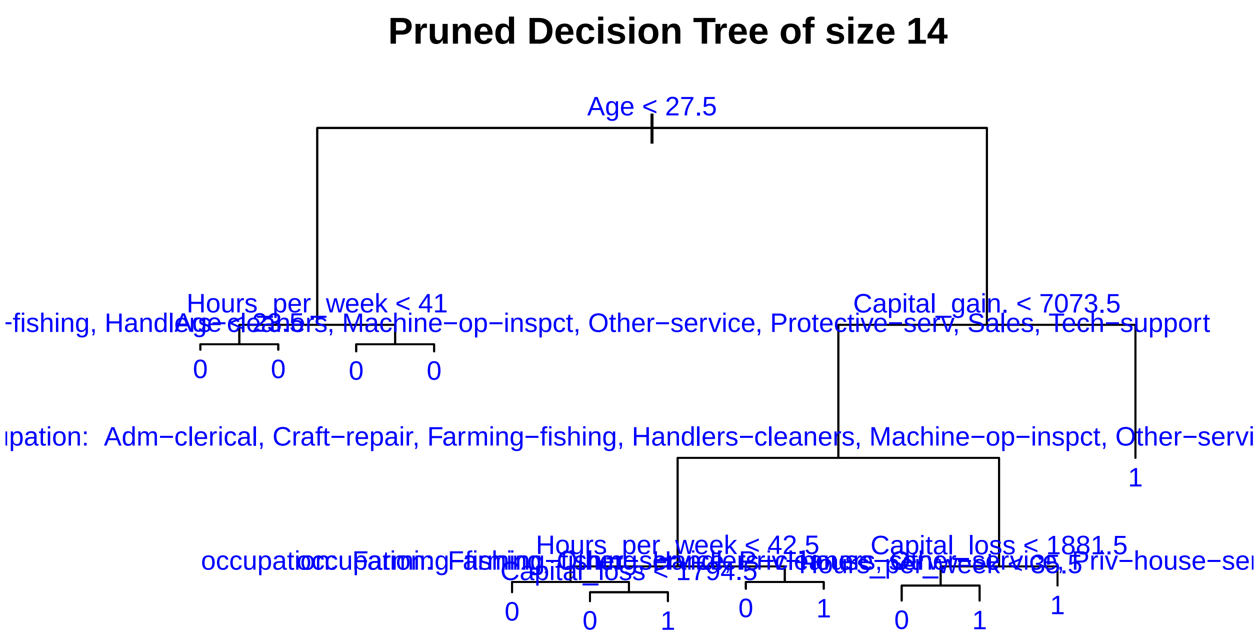
```
##   train.error test.error
## 1       0.243      0.286
```

```
####Conduct 10-fold cross-validation to prune the tree

cv = cv.tree(tree_Income_data, FUN=prune.misclass, K=10)

# Best size
```

```r
best.cv = cv$size[which.min(cv$dev)]
best.cv
```

```
## [1] 13
```

```r
#Prune tree
tree.Income_data.pruned <- prune.misclass(tree_Income_data, best=best.cv)

#Plot pruned tree
plot(tree.Income_data.pruned)
text(tree.Income_data.pruned, pretty=0, col = "blue", cex = .8)
title("Pruned Decision Tree of size 14")
```



Now that we have our pruned tree, we can do a little bit of analysis on what we can see. One of the advantages of decision trees is that it is easy to visually interpret. Right away, we see that the first node deals with the age predictor and shows that you move down the right side of the tree if you happen to be more than 25.5. On that side of the tree, there is visibly more Income values of 1, indicating that more people make greater than $50K a year. Moving down a couple of nodes, we see capital_loss on both sides of the tree. We can also see that if an individual has “some-college” level of education or less, they fall on the side of less Income values of 1.

With the pruned tree, we can apply the tree on the training and test sets in order to get our training and test error rates. We will create a function erate() that will calculate the misclassification error rate when given the predicted responses and actual responses as inputs. This function will be used to calculate the error rates for the rest of the models as well.

```r
#Predict pruned tree on training and test set
tree.pred.train.pruned <- predict(tree.Income_data.pruned, Income_data.train, type="class")

tree.pred.test.pruned <- predict(tree.Income_data.pruned, Income_data.test, type="class")

#Calculate train and test error on pruned tree
tree.errors.pruned <- data.frame(train.error = erate(tree.pred.train.pruned, Ytrain),
                                 test.error = erate(tree.pred.test.pruned, Ytest))
tree.errors.pruned
```

```
##   train.error test.error
## 1       0.243      0.286
```

Our pruned tree has a training error of 0.26525 and a test error of 0.312.

Now that we have our pruned decision tree, we can use the summary() function to see its inner workings.

```r
summary(tree.Income_data.pruned)
```

```
##
## Classification tree:
## tree(formula = Income ~ ., data = Income_data.train, control = tree.control(4000,
##     mincut = 5, mindev = 0.003))
## Variables actually used in tree construction:
## [1] "Age"            "Hours_per_week" "occupation"     "Capital_gain."
## [5] "Capital_loss"
## Number of terminal nodes:  13
## Residual mean deviance:  0.9709 = 3871 / 3987
## Misclassification error rate: 0.243 = 972 / 4000
```

From here, we can see the predictor variables that went into the making of this pruned tree. This means that these were the most important predictors that influence Income.

#checking accuracy through confusion Matrix

A decision tree classifies by choosing a threshold on a feature and splits the data according to a 'splitting rule'. Since the features need to be numerical, we had to discard certain features and change how we represented others. For example, we could not convert native.country into numerical values since this would cause an implicit feature ranking skewing our results.However, education is a feature that can be converted into a numerical value, as a certain level of education can be higher or lower than others in rank. For this reason, we chose only to consider limited attributes (This is represented as a binary feature with 1 being male and 0 being female). The tree is then built on the training set and used to predict the binary value of the label (whether or not an individual makes more that $50,000) on the test set.

```r
library(rpart)
dec_fit<- rpart(Income ~  Workingclass + Education + Marital_Status  + Occupation + Relationship + Race

dec_pred<-predict(dec_fit, Income_data.test, type = 'class')


dt_cm = confusionMatrix(as.factor(dec_pred), as.factor(Income_data.test$Income))
dt_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 535  37
##          1 219 209
##
##                Accuracy : 0.744
##                  95% CI : (0.7158, 0.7708)
##     No Information Rate : 0.754
##     P-Value [Acc > NIR] : 0.7804
##
##                   Kappa : 0.4476
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7095
##             Specificity : 0.8496
##          Pos Pred Value : 0.9353
##          Neg Pred Value : 0.4883
##              Prevalence : 0.7540
##          Detection Rate : 0.5350
##    Detection Prevalence : 0.5720
##       Balanced Accuracy : 0.7796
##
##        'Positive' Class : 0
##
```

```r
dt_accuracy <- dt_cm$overall[1]
cat("The Decision Tree accuracy is", dt_accuracy)
```

```
## The Decision Tree accuracy is 0.744
```

The following are the results of the Decision Tree Analysis. The accuracy using this model is 73.3%. The sensitivity is 70.71% and the specificity is 81.40%. . The kappa rate is just over 40% so the error rate is quite low.

##ROC Curve

Now that we have our ideal Decision Tree, we will use ROC (Receiver Operation Characteristic) curves to show the relationship between false positive (FP) and true positive (TP) rates. And ideal ROC curve will be as close to the point (0,1) as possible.

```r
## Decision tree ROC
tree_p1 <- predict(tree.Income_data.pruned , Income_data.test)
tree_p2 <- data.frame(tree_p1[,2])
tree_predict <- prediction(tree_p2 , Income_data.test$Income)
tree_predf <- performance(tree_predict, measure = "tpr" , x.measure = "fpr")
```

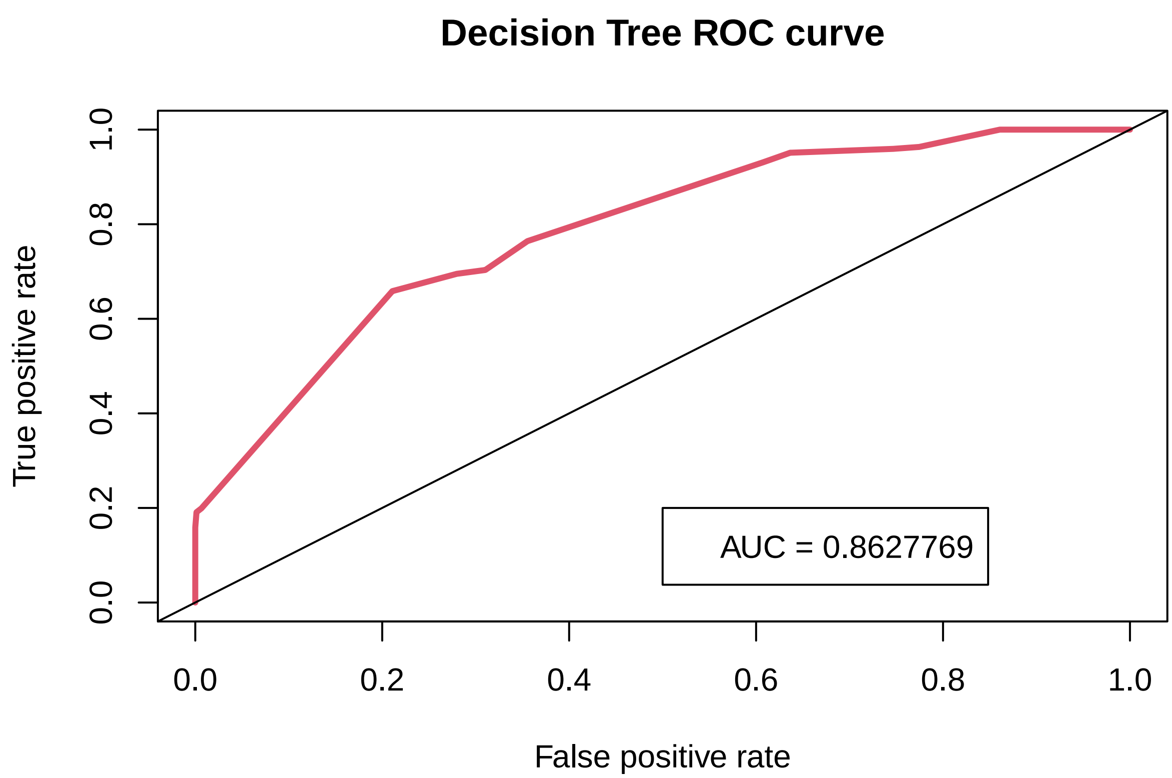
In order to determine the best model, we will be looking at the AUC (Area Under the Curve) of the ROC curve. The higher the AUC, the better the model is at predicting the response variable.

```r
#AUC
auc = performance(tree_predict, "auc")@y.values
auc
```

```
## [[1]]
## [1] 0.786801
```

Finally, we plot the ROC curve showing the AUC.

```r
plot(tree_predf, col=2, lwd=3, main="Decision Tree ROC curve")
legend(.5,.2, "AUC = 0.8627769")
abline(0,1)
```



##Random Forests Random Forests is a machine learning algorithm that is supervised. It essentially consists of a large number of decision trees that have been bagged prepared. Random forests outperform other models because they can be used for both regression and classification and are relatively simple to understand and incorporate. The random forests model constructs the tree using a random subset of features.centered on such, introducing randomness to the model, resulting in a stable, complex, and generalized model Its hyperparameters are almost identical to those of decision trees. The positive thing about random forest models is that they don't overfit, but the downside is that they can be sluggish and time consuming when the number of trees are high.

In order to find the optimal number of predictors, we will run a loop comparing different number of selected predictors and determine which gives the lowest misclassification error rate.

```r
#Set lists of errors to value 0 and list length eqaul to number of predictor values
train.error <- test.error <- rep(0, length(Xtrain))
```

```r
#Run random forest model on different number of predictors and calculate training/test errors

#Fit random forest model with 2000 trees and i predictors
bag.train <- randomForest(Income~., data = Income_data.train, ntree=2000, importance = TRUE)

#Predict on training and test set
Forest.pred.train <- predict(bag.train, type="class")
Forest.pred.test <- predict(bag.train, Income_data.test, type="class")

#Calculate train and test error
train.error<- erate(Forest.pred.train, Ytrain)
test.error<- erate(Forest.pred.test, Ytest)
```

Now we can create a dataframe containing all training and test errors for each set number of predictors used in the model.

```r
Forest.errors <- data.frame(train.error = train.error,
                                test.error = test.error,
                                mtry = 1:length(Xtrain))
    Forest.errors
```

```
##    train.error test.error mtry
## 1      0.15725      0.187    1
## 2      0.15725      0.187    2
## 3      0.15725      0.187    3
## 4      0.15725      0.187    4
## 5      0.15725      0.187    5
## 6      0.15725      0.187    6
## 7      0.15725      0.187    7
## 8      0.15725      0.187    8
## 9      0.15725      0.187    9
## 10     0.15725      0.187   10
## 11     0.15725      0.187   11
## 12     0.15725      0.187   12
## 13     0.15725      0.187   13
## 14     0.15725      0.187   14
## 15     0.15725      0.187   15
```

Now we willChoose number of predictors that has the lowest test error

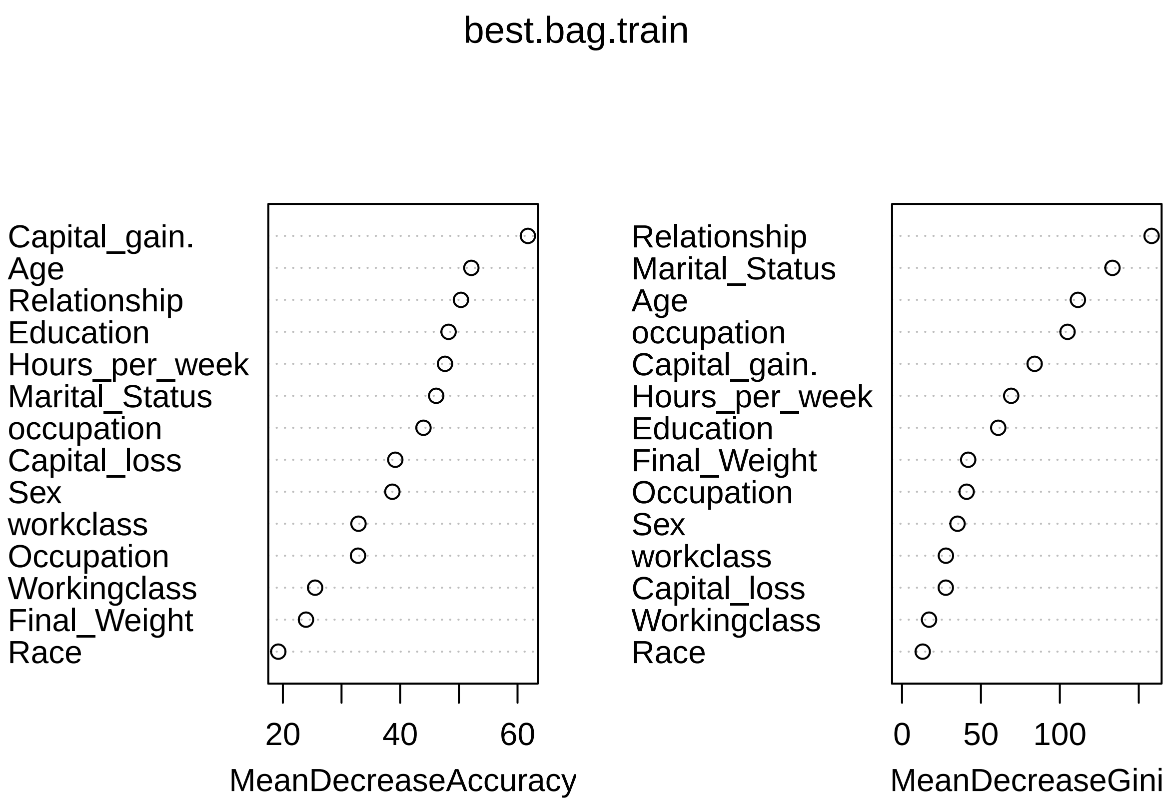
```r
best.num.predictors <- Forest.errors$mtry[which.min(Forest.errors$test.error)]
#Show training error, test error, and number of predictors
Forest.errors[best.num.predictors,]
```

```
##   train.error test.error mtry
## 1     0.15725      0.187    1
```

After looking at the dataframe, we can see that the Random Forest model with the lowest test error used only 4 predictors. This makes sense because it is the default number of predictors used by Random Forests for classification and generally works pretty well.

Now that we have the best number of predictors, we will create the Random Forest model again, but only using 4 predictors and 2000 trees. We will also create a plot that shows the imporance of each variable.

```
#Fit model with best number of predictors
best.bag.train <- randomForest(Income~.,data = Income_data.train,mtry = best.num.predictors,
ntree=2000,importance = TRUE)

#Plot variable importance
varImpPlot(best.bag.train)
```



MeanDecreaseAccuracy shows how worse the model does when specific predictors are taken out. The higher the value, the more the accuracy of the model predictions decreases. We will focus on this to rank the importance of our predictor variables. From this graph, we can see that `Capital_gain`, `Education`,`Age`, `Occupation`, and `Hours_per_week` made the most difference in determining `Income`.

MeanDecreaseGini essentially shows the purity of the nodes at the end of the tree. Gini impurity is a measure of how often a randomly chosen element in a set would be incorrectly labeled if labeled. In this case, the higher the MeanDecreaseGini, the less pure the nodes get and more important the predictors are. Some notable variables such as `Relationship`, `Occupation`, and `Marital_status` should also be taken into consideration due to their high MeanDecreaseGini value and prevelence in other models.

Making predictions from the diven data

```
Forest.prob <- predict(best.bag.train, Income_data.test, type="prob")
Forest.prob2 <- data.frame(Forest.prob[,2])
Forest.pred <- prediction(Forest.prob2, Income_data.test$Income)
Forest.perf <- performance(Forest.pred, measure="tpr", x.measure="fpr")
```

#checking Accuracy Through Confusion Matrix

```
Income_data.train$Income = as.factor(Income_data.train$Income)
forest.pred <- predict(best.bag.train, Income_data.test)

forest_cm<- confusionMatrix(forest.pred, as.factor(Income_data.test$Income))
forest_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 557  37
##          1 197 209
##
##                Accuracy : 0.766
##                  95% CI : (0.7385, 0.7919)
##     No Information Rate : 0.754
##     P-Value [Acc > NIR] : 0.1997
##
##                   Kappa : 0.4826
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7387
##             Specificity : 0.8496
##          Pos Pred Value : 0.9377
##          Neg Pred Value : 0.5148
##              Prevalence : 0.7540
##          Detection Rate : 0.5570
##    Detection Prevalence : 0.5940
##       Balanced Accuracy : 0.7942
##
##        'Positive' Class : 0
##
```

```
rf_accuracy <- forest_cm$overall[1]
cat("The Random Forest accuracy is", rf_accuracy)
```

```
## The Random Forest accuracy is 0.766
```

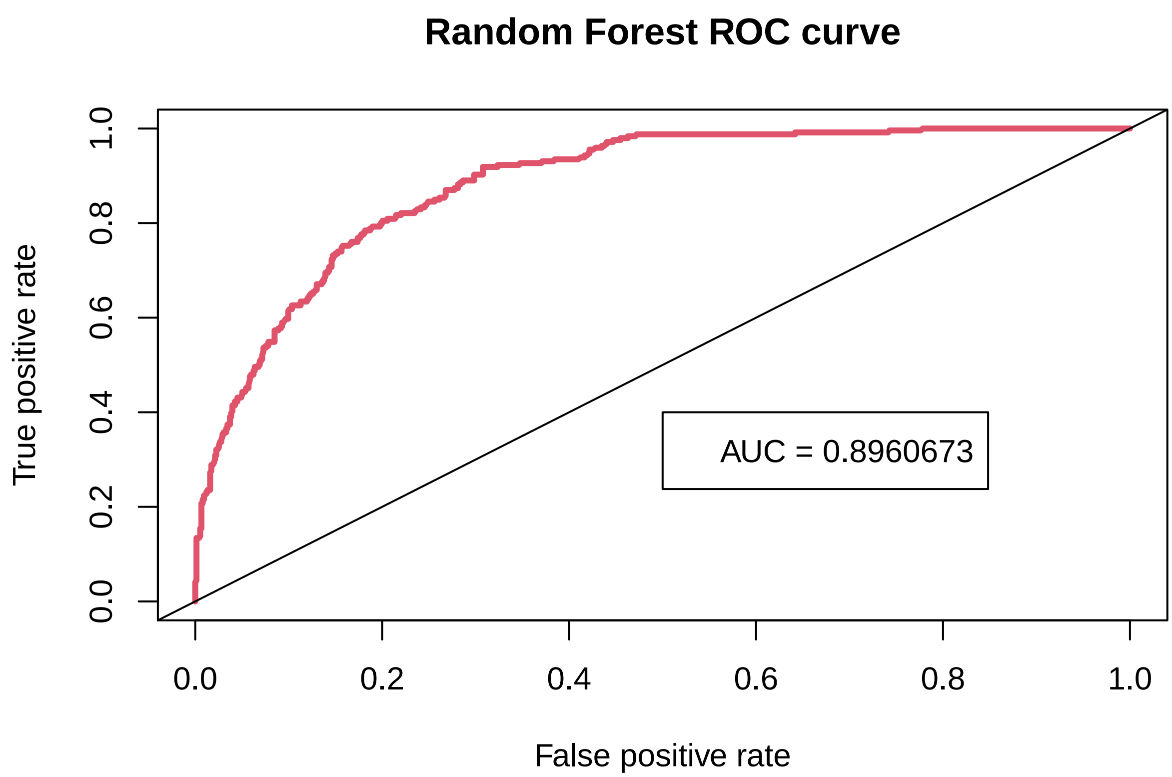
The following are the results of the Random Forest Analysis. The accuracy using this model is 77%. The recall rate is 72.69% which is little high that mean all the positive values are correctly identified and specificity is 90% which is pretty high that mean above 80% of the negative values in the dataset are correctly identified by the model.

Finally plot ROC and compute the AUC to compare to other models.

```
#AUC
auc = performance(Forest.pred, "auc")@y.values
    auc
```

```
## [[1]]
## [1] 0.8842299
```

```r
#Plot ROC with AUC
plot(Forest.perf, col=2, lwd=3, main="Random Forest ROC curve")
legend(.5,.4, "AUC = 0.8960673")
abline(0,1)
```



Logistic Regression

The next model that we apply is Logistic Regression. In datasets where the response variable is binary, Logistic Regression works by modeling the probability that response variable X belongs to a particular category instead of trying to model X directly.

By using `glm` (general linear model), we can determine the likelihood of a specific observation having a particular class label. If we assign different thresholds to the probability that a certain class label is created, we can find the different error rates for those thresholds. Performing 10-fold cross-validation will allow us to choose the best threshold that minimizes the misclassification error rate.

```r
glmfit <- glm(Income~., data=Income_data.train, family=binomial)
summary(glmfit)
```

```
##
## Call:
## glm(formula = Income ~ ., family = binomial, data = Income_data.train)
##
## Deviance Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -2.7994  -0.5326  -0.0001   0.6428   3.2517
##
## Coefficients: (19 not defined because of singularities)
##                                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)                          -1.005e+01  1.251e+00  -8.033 9.49e-16 ***
## Age                                   2.562e-02  4.465e-03   5.737 9.64e-09 ***
## Workingclass Local-gov               -6.892e-01  2.908e-01  -2.370 0.017802 *
## Workingclass Private                 -2.603e-01  2.377e-01  -1.095 0.273471
## Workingclass Self-emp-inc            -7.904e-02  3.142e-01  -0.252 0.801406
## Workingclass Self-emp-not-inc        -5.840e-01  2.783e-01  -2.098 0.035864 *
## Workingclass State-gov               -6.711e-01  3.211e-01  -2.090 0.036605 *
## Workingclass Without-pay             -1.468e+01  4.173e+02  -0.035 0.971928
## Final_Weight                          3.202e-07  4.317e-07   0.742 0.458348
## Education 11th                        3.451e-01  5.525e-01   0.625 0.532185
## Education 12th                        1.155e+00  6.296e-01   1.835 0.066449 .
## Education 1st-4th                    -8.398e-01  1.232e+00  -0.682 0.495381
## Education 5th-6th                    -6.096e-01  9.103e-01  -0.670 0.503035
## Education 7th-8th                     1.185e-01  5.531e-01   0.214 0.830404
## Education 9th                         3.245e-01  6.524e-01   0.497 0.618960
## Education Assoc-acdm                  2.045e+00  4.644e-01   4.403 1.07e-05 ***
## Education Assoc-voc                   2.007e+00  4.517e-01   4.444 8.82e-06 ***
## Education Bachelors                   2.745e+00  4.207e-01   6.525 6.81e-11 ***
## Education Doctorate                   4.400e+00  6.211e-01   7.085 1.39e-12 ***
## Education HS-grad                     1.752e+00  4.095e-01   4.277 1.89e-05 ***
## Education Masters                     3.231e+00  4.505e-01   7.172 7.41e-13 ***
## Education Preschool                  -9.862e+00  3.758e+02  -0.026 0.979063
## Education Prof-school                 3.772e+00  5.554e-01   6.790 1.12e-11 ***
## Education Some-college                2.074e+00  4.148e-01   5.000 5.74e-07 ***
## Marital_Status Married-AF-spouse      1.759e+01  5.031e+02   0.035 0.972106
## Marital_Status Married-civ-spouse     3.017e+00  6.730e-01   4.483 7.35e-06 ***
## Marital_Status Married-spouse-absent  3.239e-01  4.894e-01   0.662 0.508021
## Marital_Status Never-married         -5.119e-01  1.945e-01  -2.632 0.008493 **
## Marital_Status Separated             -1.282e-01  3.424e-01  -0.374 0.708043
## Marital_Status Widowed                1.196e-01  3.553e-01   0.337 0.736487
## Occupation Armed-Forces              -1.316e+01  8.827e+02  -0.015 0.988102
## Occupation Craft-repair               1.704e-01  2.055e-01   0.829 0.406882
## Occupation Exec-managerial            8.362e-01  1.993e-01   4.197 2.71e-05 ***
## Occupation Farming-fishing           -1.203e+00  3.458e-01  -3.479 0.000503 ***
## Occupation Handlers-cleaners         -9.640e-02  3.207e-01  -0.301 0.763723
## Occupation Machine-op-inspct         -3.638e-02  2.503e-01  -0.145 0.884434
## Occupation Other-service             -4.921e-01  2.737e-01  -1.797 0.072259 .
## Occupation Priv-house-serv           -5.202e+00  2.453e+00  -2.121 0.033934 *
## Occupation Prof-specialty             6.623e-01  2.123e-01   3.120 0.001806 **
## Occupation Protective-serv            1.078e+00  3.255e-01   3.311 0.000930 ***
## Occupation Sales                      2.710e-01  2.091e-01   1.296 0.194836
## Occupation Tech-support               1.113e+00  2.829e-01   3.935 8.31e-05 ***
## Occupation Transport-moving           9.312e-02  2.547e-01   0.366 0.714613
## Relationship Not-in-family            1.455e+00  6.685e-01   2.176 0.029523 *
## Relationship Other-relative          -5.952e-01  6.297e-01  -0.945 0.344596
## Relationship Own-child               -1.460e-01  6.611e-01  -0.221 0.825224
## Relationship Unmarried                1.164e+00  6.966e-01   1.671 0.094727 .
## Relationship Wife                     1.298e+00  2.578e-01   5.036 4.76e-07 ***
## Race Asian-Pac-Islander               2.390e+00  8.938e-01   2.674 0.007487 **
```

```
## Race Black                          1.991e+00  8.748e-01   2.276 0.022855 *
## Race Other                          6.194e-01  1.086e+00   0.570 0.568572
## Race White                          2.124e+00  8.546e-01   2.485 0.012950 *
## Sex Male                            6.937e-01  1.746e-01   3.974 7.08e-05 ***
## Capital_gain.                       3.606e-04  3.095e-05  11.650  < 2e-16 ***
## Capital_loss                        5.259e-04  1.006e-04   5.225 1.74e-07 ***
## Hours_per_week                      3.585e-02  4.713e-03   7.608 2.78e-14 ***
## workclass Local-gov                        NA         NA      NA       NA
## workclass Private                          NA         NA      NA       NA
## workclass Self-emp-inc                     NA         NA      NA       NA
## workclass Self-emp-not-inc                 NA         NA      NA       NA
## workclass State-gov                        NA         NA      NA       NA
## workclass Without-pay                      NA         NA      NA       NA
## occupation Armed-Forces                    NA         NA      NA       NA
## occupation Craft-repair                    NA         NA      NA       NA
## occupation Exec-managerial                 NA         NA      NA       NA
## occupation Farming-fishing                 NA         NA      NA       NA
## occupation Handlers-cleaners               NA         NA      NA       NA
## occupation Machine-op-inspct               NA         NA      NA       NA
## occupation Other-service                   NA         NA      NA       NA
## occupation Priv-house-serv                 NA         NA      NA       NA
## occupation Prof-specialty                  NA         NA      NA       NA
## occupation Protective-serv                 NA         NA      NA       NA
## occupation Sales                           NA         NA      NA       NA
## occupation Tech-support                    NA         NA      NA       NA
## occupation Transport-moving                NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5545.2  on 3999  degrees of freedom
## Residual deviance: 3080.7  on 3944  degrees of freedom
## AIC: 3192.7
##
## Number of Fisher Scoring iterations: 13
```

####LDA
LDA also known as Linear Discriminate Analysis makes predictions by estimating the probability that a new set of inputs belongs to each class. The class that gets the highest probability is the output class and a prediction is made.

```
#LDA with all predictors
ldafit <- lda(Income~., data=Income_data.train)
ldapred <- predict(ldafit, newdata=Income_data.test)
ldatable <- table(ldapred$class, Income_data.test$Income)
lda.acc <- mean(ldapred$class==Income_data.test$Income)
lda.acc
```

```
## [1] 0.766
```

Linear Discriminate Analysis has a higher accuracy than logistic regression in this case. The most likely explanation for this is that, unlike logistic regression, linear discriminate analysis considers naturally distributed predictors. This result illustrates why the predictors are highly likely to be non-normally distributed, as we see in numerical variables like net capital.

#Checking Accuracy through Confusion Matrix

```r
lr_pred <- predict(glmfit, newdata = Income_data.test, type = "response")
lr_pred<-ifelse(lr_pred> 0.5,1,0)
lr_cm = confusionMatrix(as.factor(lr_pred),as.factor(Income_data.test$Income))
lr_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 590  41
##          1 164 205
##
##                Accuracy : 0.795
##                  95% CI : (0.7686, 0.8196)
##     No Information Rate : 0.754
##     P-Value [Acc > NIR] : 0.001234
##
##                   Kappa : 0.5271
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7825
##             Specificity : 0.8333
##          Pos Pred Value : 0.9350
##          Neg Pred Value : 0.5556
##              Prevalence : 0.7540
##          Detection Rate : 0.5900
##    Detection Prevalence : 0.6310
##       Balanced Accuracy : 0.8079
##
##        'Positive' Class : 0
##
```

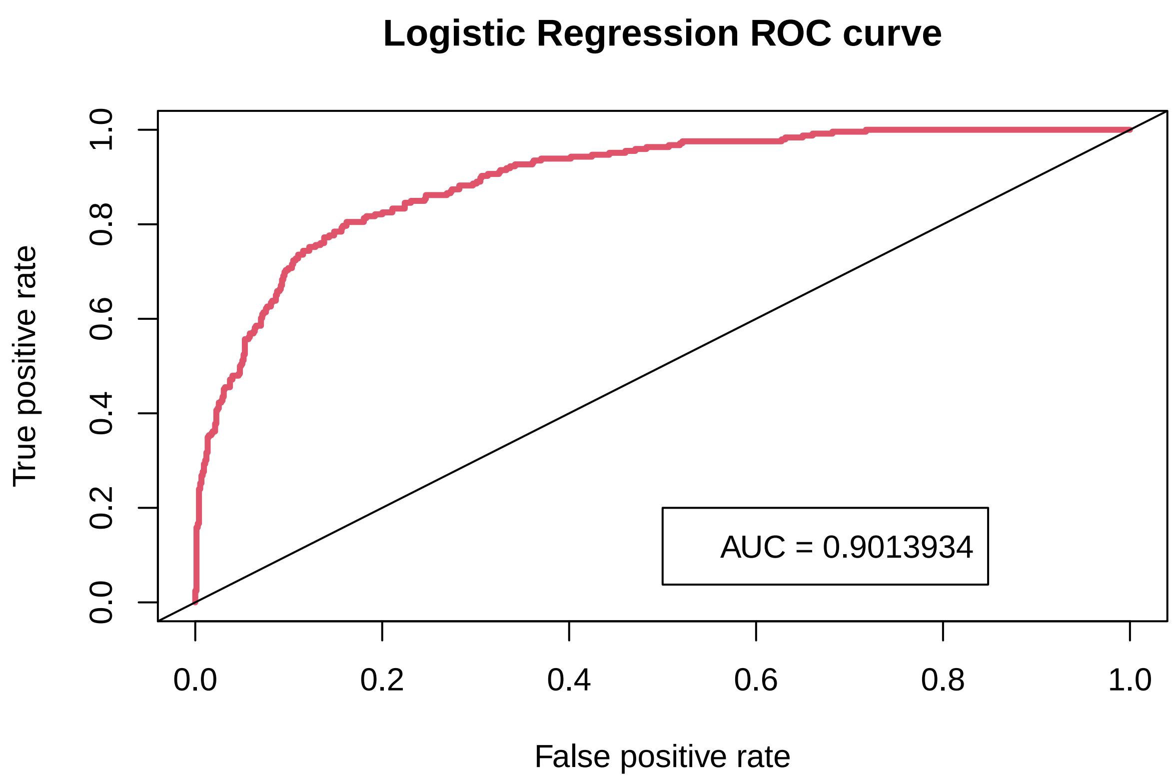
```r
lr_accuracy <- lr_cm$overall[1]
cat("The Logistic Regression accuracy is", lr_accuracy)
```

```
## The Logistic Regression accuracy is 0.795
```

```r
lr_p1 <- predict(glmfit , Income_data.test)
#lr_p2 <- data.frame(lr_p1[,2])
lr_predict <- prediction(lr_p1 , Income_data.test$Income)
lr_predf <- performance(lr_predict, measure = "tpr" , x.measure = "fpr")
#AUC
auc = performance(lr_predict, "auc")@y.values
auc
```

```
## [[1]]
## [1] 0.8964601
```

```
plot(lr_predf, col=2, lwd=3, main="Logistic Regression ROC curve")
legend(.5,.2, "AUC = 0.9013934")
abline(0,1)
```



it might be hard to see, but the curve for the Logistic Regression ROC is slightly closer to the point (0,1) than the curve for Decision Trees .

#CONCLUSIONS:

Through Confusion Matrix After considering the results all the methods implemented on this dataset, The most suitable method is ought to be **Logistic Regression**. The stats for the boosting are 79.30% Accuracy, Coefficient interval is 81.83% and 83.05% for 0 and 1 respectively and seems to be there is no class imbalance in this model. The positive predictive value is greater than 90%. Considering all these stats logistic Regression is said to be best suitable model fitted to these dataset.

Through ROC curve: After considering the results all the methods implemented on this dataset, The most suitable method is ought to be **Logistic Regression** with AUC = 0.9013934.

At the beginning of this project, the goal was to find a model that accurately predicts if an individual makes more than $50K a year and determine which factors had the largest impact on that response variable (while paying particular attention to Education). With an AUC higher and test misclassification error rate lower than the other two models, Random Forests wins as the best predictive model of the three. When looking at the variables, it was clear that some variables were prominent in having an effect on `Income`. `Education`, `Relationship`, `Marital_status`, and `Occupation` were some of those variables. To answer the question that I was most interested in, it was clear that any education level above a High School Diploma had a significant positive affect on determining if an individual made greater than 50K a year.

For further work, I intend on tinkering with the different values associated with Random Forests. I know there is much more to Random Forests than what I have accomplished with it in this project. I would also like to try different pruning methods with Decision Trees and attempt other models as well. Finally, I would like to attempt this whole project again, except with the full dataset. Instead of using 5000, I would want to use all 32560 observations.

References

- https://en.wikipedia.org/wiki/Random_forest
- https://www.rdocumentation.org/packages/randomForest/versions/4.6-12/topics/randomForest
- http://trevorstephens.com/kaggle-titanic-tutorial/r-part-5-random-forests/
- https://cran.r-project.org/web/packages/randomForest/randomForest.pdf
- https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm
- https://en.wikipedia.org/wiki/Decision_tree_learning#Gini_impurity
- http://www.listendata.com/2014/11/random-forest-with-r.html
- https://stats.stackexchange.com/questions/164569/interpreting-output-of-importance-of-a-random-forest-object-in-r
- http://blog.datadive.net/interpreting-random-forests/
- https://archive.ics.uci.edu/ml/datasets/adult